6.886 talk: Engineering a cache-oblivious sorting algorithm

Brodal, Fagerberg, Vinther (2008)

Sualeh Asif

MIT Computer Science Artificial Intelligence Laboratory
Massachusetts Institute of Technology



Cache-Oblivious Sorting

Cache-Oblivious Sorting: A Timeline

Funnel Sorting
(Frigo et.al 1999)

Cache-Oblivious Sorting: A Timeline

Funnel Sorting
(Frigo et.al ; 1999)



Lazy Funnel Sorting
(Brodal, Fagerberg ; 2002)

Cache-Oblivious Sorting: A Timeline

Funnel Sorting
(Frigo et.al ; 1999)

↓

Lazy Funnel Sorting
(Brodal, Fagerberg ; 2002)

↓

Engineered Cache-Oblivious Sort
(Brodal, Fagerberg, Vinther ; 2008)

Cache-Oblivious Sorting: A Timeline

Funnel Sorting
(Frigo et.al ; 1999)

↓

Lazy Funnel Sorting
(Brodal, Fagerberg ; 2002)

↓

Engineered Cache-Oblivious Sort
(Brodal, Fagerberg, Vinther ; 2008)

→ **Thm 1 (Cache-Oblivious Sorting)**

N items, M (small) memory
B blocks.

Funnel sorting can sort the N items in time:

$$\Theta\left(\frac{N}{B} \log_{M/B} N/B\right)$$

This is optimal under reasonable assumptions.

Cache-Oblivious Sorting

Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory

B blocks.

Funnel sorting time:

$$\Theta\left(\frac{N}{B} \log_{M/B} N/B\right)$$

Cache-Oblivious Sorting

Thm 1 (Cache-Oblivious Sorting)
N items, M (small) memory
B blocks.
Funnel sorting time:

$$\Theta\left(\frac{N}{B} \log_{M/B} N/B\right)$$

Why Care ...?

Cache-Oblivious Sorting

Why Care ...?

* Sort is (almost surely!!) fundamental problem
  ⇒ fast sorting implementation independant of (constantly) changing memory hierarchy!

Thm 1 (Cache-Oblivious Sorting)
N items, M (small) memory
B blocks.
Funnel sorting time:

$$\Theta\left(\frac{N}{B} \log_{M/B} N/B\right)$$

Cache-Oblivious Sorting

Thm 1 (Cache-Oblivious Sorting)
N items, M (small) memory
B blocks.
Funnel sorting time:

$$\Theta\left(\frac{N}{B} \log_{M/B} \frac{N}{B}\right)$$

Why Care ...?

* Sort is (almost surely!!) fundamental problem
  $\Rightarrow$ fast sorting implementation independant of (constantly) changing memory hierarchy!

* Experimentally check which heuristics are important

Cache-Oblivious Sorting

Thm 1 (Cache-Oblivious Sorting)
N items, M (small) memory
B blocks.
Funnel sorting time:

$$\Theta\left(\frac{N}{B} \log_{M/B} N/B\right)$$

Why Care ...?

* Sort is (almost surely!!) fundamental problem
  $\Rightarrow$ fast sorting implementation independant of (constantly) changing memory hierarchy!
* Experimentally check which heuristics are important
* Curious humans ... and it is cool!

Cache-Oblivious Sorting

Thm 1 (Cache-Oblivious Sorting)
N items, M (small) memory
B blocks.
Funnel sorting time:

$$\Theta\left(\frac{N}{B} \log_{M/B} N/B\right)$$

Why Care ...?

* Sort is (almost surely!!) fundamental problem
  $\Rightarrow$ fast sorting implementation independant of (constantly) changing memory hierarchy!
* Experimentally check which heuristics are important
* Curious humans ... and it is cool!

Punch line: near the best regardless !!

(Top 3) ← near the best

→ (disk vs ram vs mixed) regardless

Cache-Oblivious Sorting

Thm 1 (Cache-Oblivious Sorting)
N items, M (small) memory
B blocks.
Funnel sorting time:

$$\Theta\left(\frac{N}{B} \log_{M/B} \frac{N}{B}\right)$$

Why Care ...?

* Sort is (almost surely!!) fundamental problem
  $\Rightarrow$ fast sorting implementation independant of (constantly) changing memory hierarchy!
* Experimentally check which heuristics are important
* Curious humans ... and it is cool!

Questions?

FUNNEL - SORTING : A RECAP

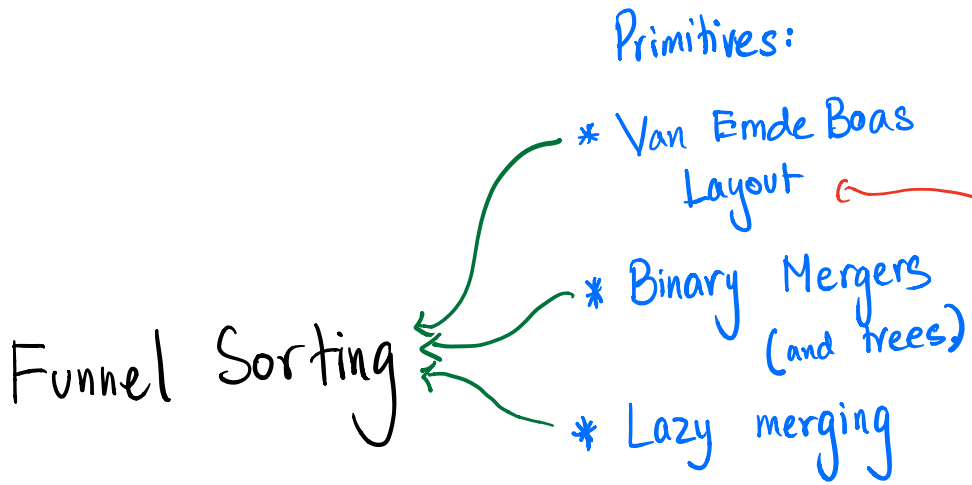
Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory

B blocks.

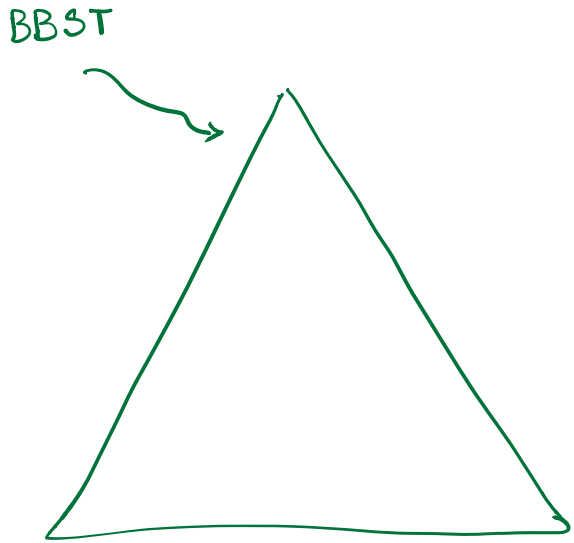
Funnel sorting time:

$$\Theta\left(\frac{N}{B} \log_{M/B} N/B\right)$$



FUNNEL - SORTING : A RECAP

Van Emde Boas Layout



Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory

B blocks.

Funnel sorting time:

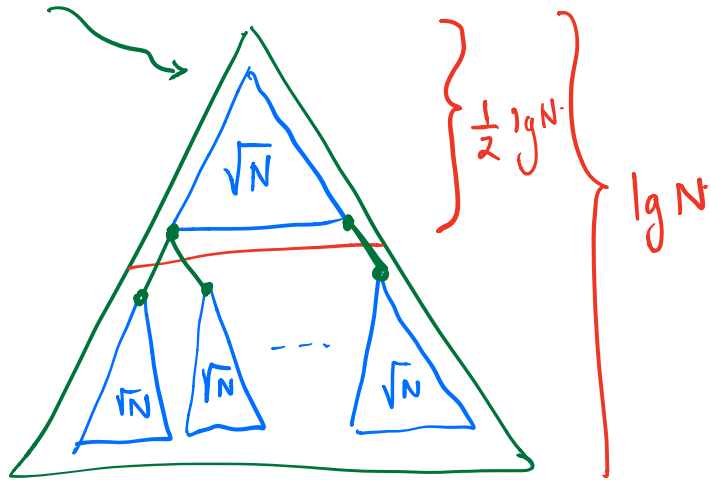
$$\Theta\left(\frac{N}{B} \log_{M/B} N/B\right)$$

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

FUNNEL - SORTING : A RECAP

Van Emde Boas Layout

BBST - N elements



Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory, B blocks.

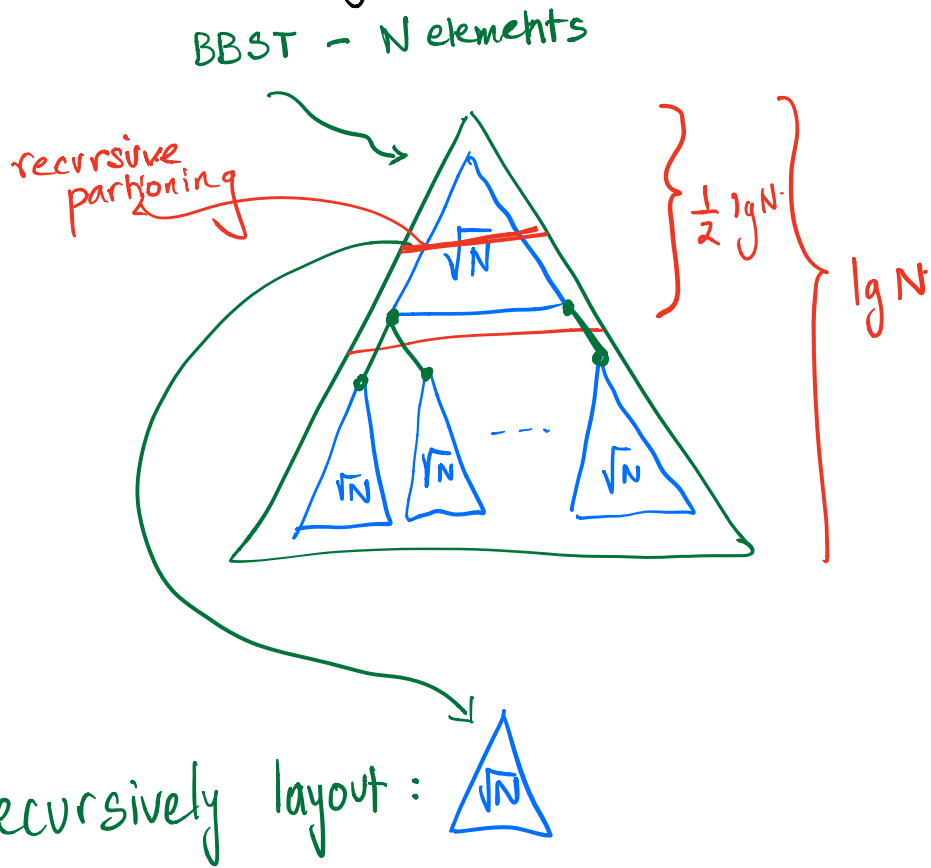
Funnel sorting time:

$$\Theta\left(\frac{N}{B} \log_{M/B} N/B\right)$$

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

FUNNEL - SORTING : A RECAP

Van Emde Boas Layout



Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory, B blocks.

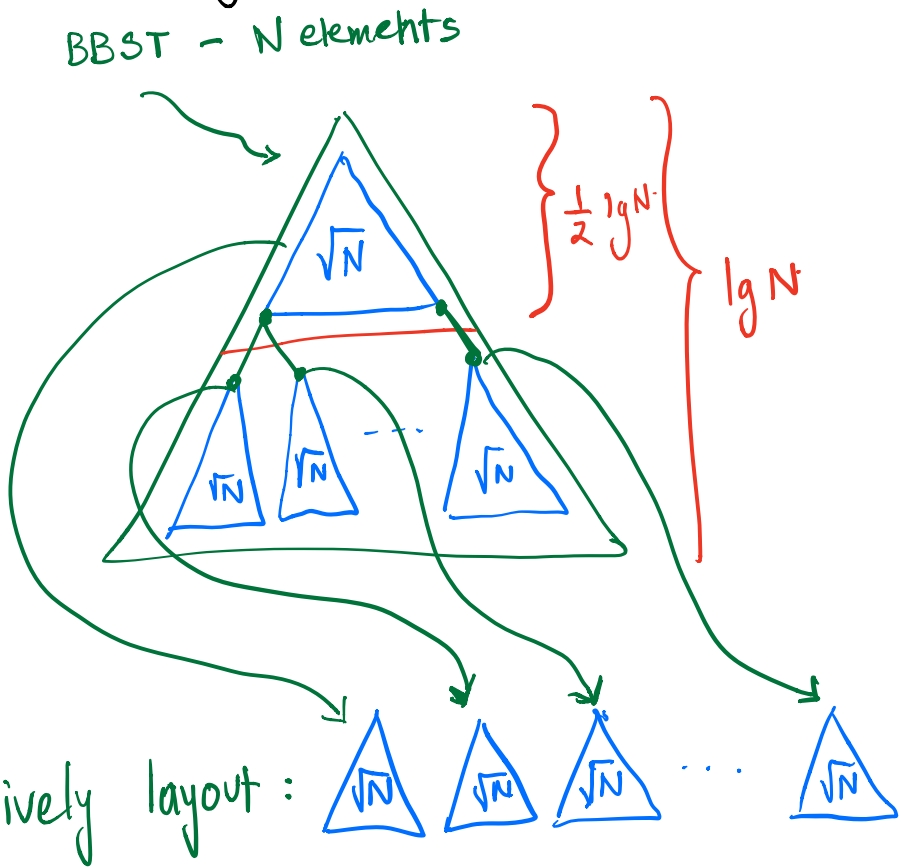
Funnel sorting time:

$$\Theta\left(\frac{N}{B} \log_{M/B} {}^{N}/_{B}\right)$$

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

FUNNEL - SORTING : A RECAP

Van Emde Boas Layout



Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory, B blocks.

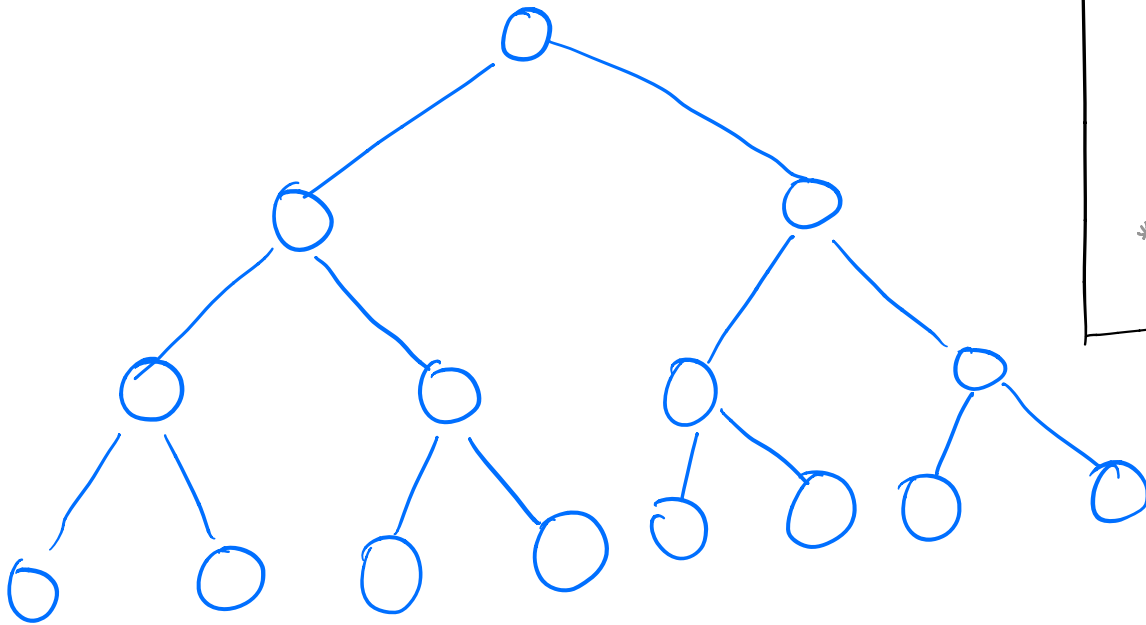
Funnel sorting time:

$$\Theta\left(\frac{N}{B}\log_{M/B}{}^{N}/_{B}\right)$$

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

FUNNEL - SORTING : A RECAP

Van Emde Boas Layout



Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory

B blocks.

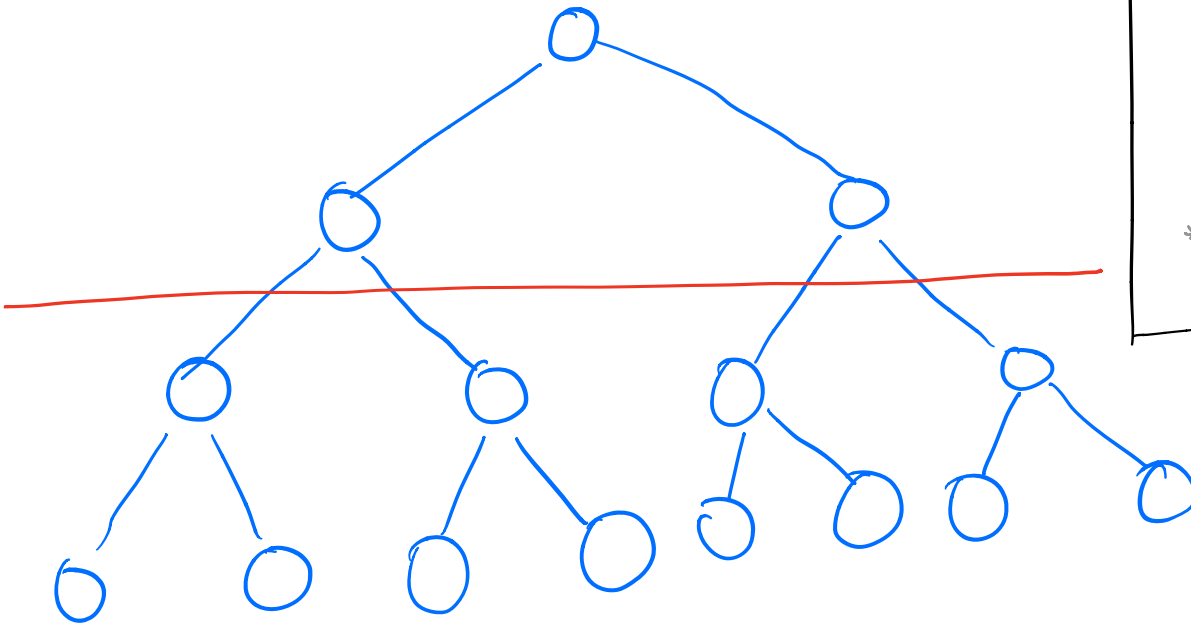
Funnel sorting time:

$$\Theta\left(\frac{N}{B} \log_{M/B} N/B\right)$$

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

FUNNEL - SORTING : A RECAP

Van Emde Boas Layout



Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory

B blocks.

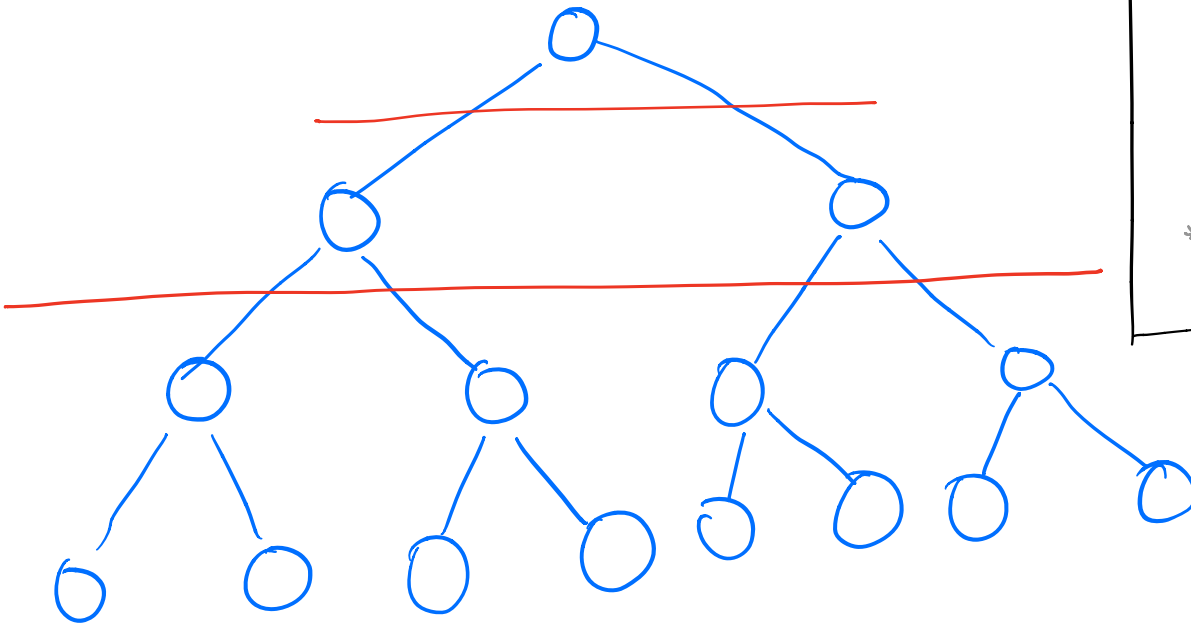
Funnel sorting time:

$$\Theta\left(\frac{N}{B} \log_{M/B} N/B\right)$$

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

FUNNEL - SORTING : A RECAP

Van Emde Boas Layout



Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory

B blocks.

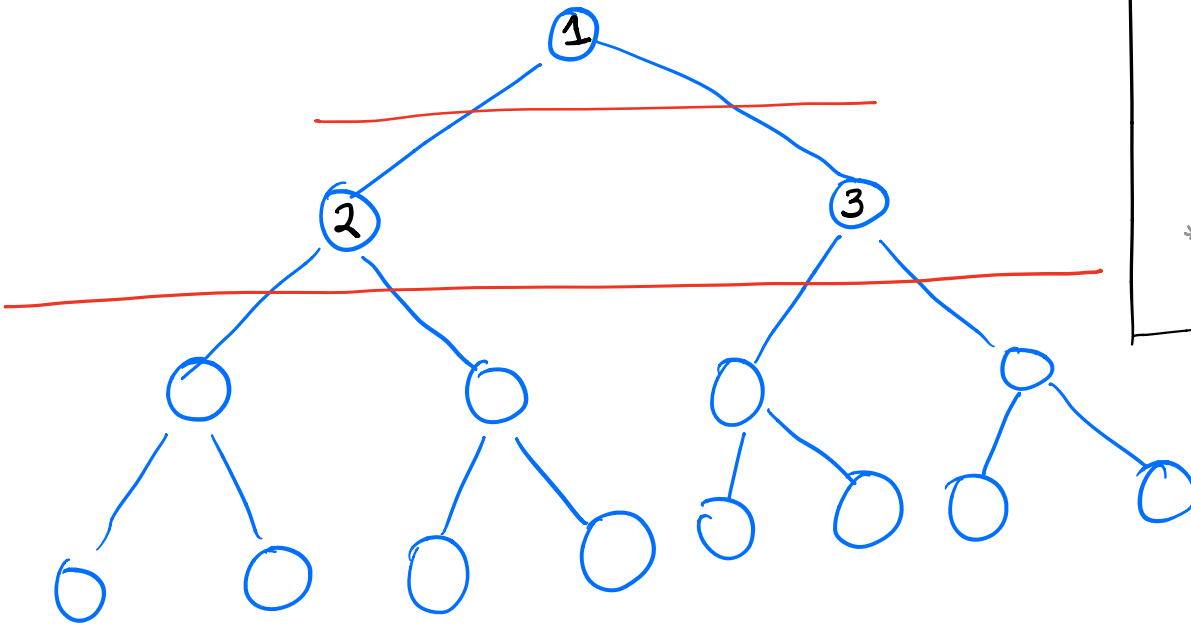
Funnel sorting time:

$$\Theta\left(\frac{N}{B} \log_{M/B} N/B\right)$$

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

FUNNEL - SORTING : A RECAP

Van Emde Boas Layout



Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory

B blocks.

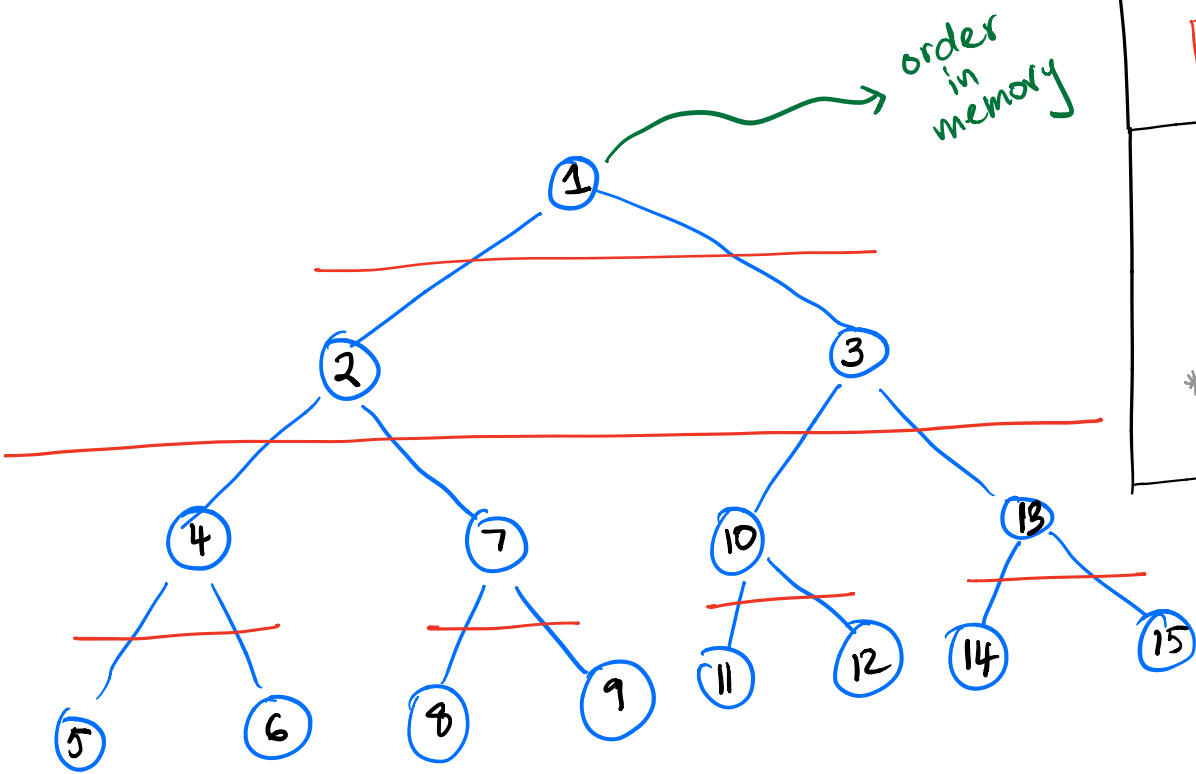
Funnel sorting time:

$$\Theta\left(\frac{N}{B} \log_{M/B} N/B\right)$$

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

FUNNEL - SORTING : A RECAP

Van Emde Boas Layout



Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory, B blocks.

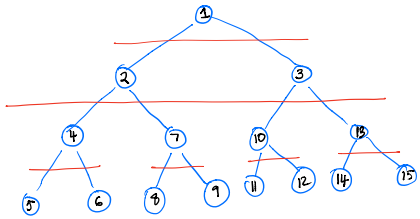
Funnel sorting time:

$$\Theta\left(\frac{N}{B} \log_{M/B} N/B\right)$$

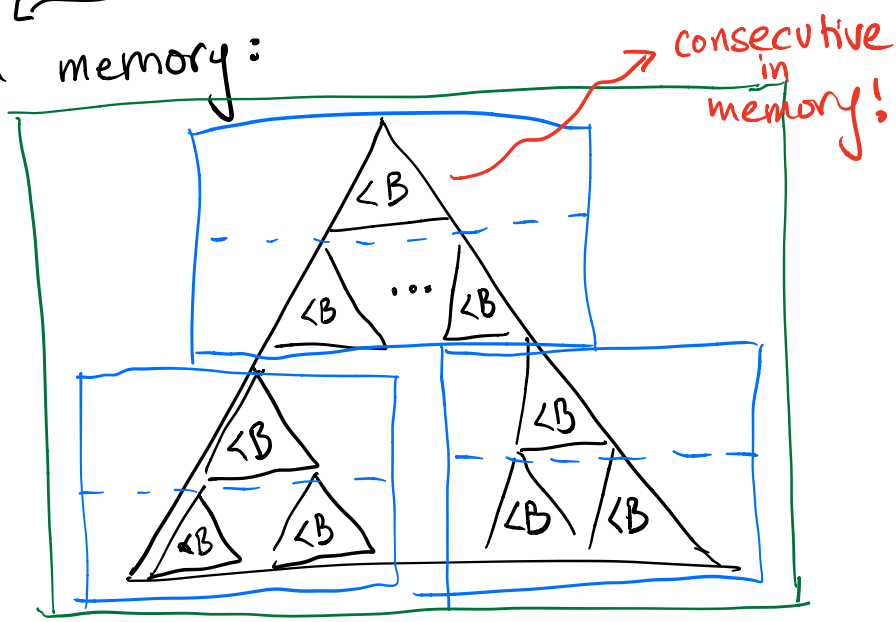
* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

FUNNEL - SORTING : A RECAP

Van Emde Boas Layout



fast in memory:



Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory, B blocks.

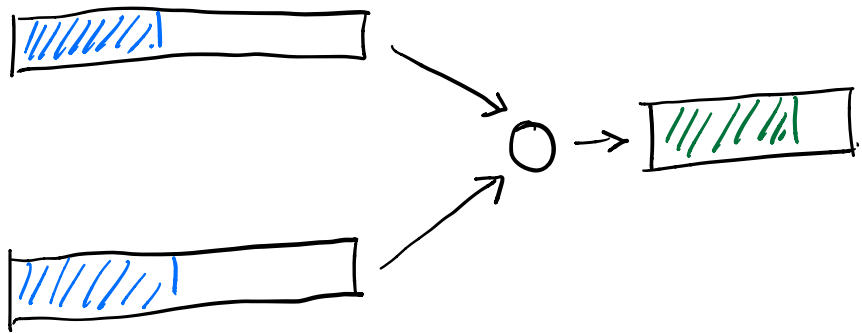
Funnel sorting time:

$$\Theta\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$$

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

FUNNEL - SORTING : A RECAP

Binary Merging



Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory

B blocks.

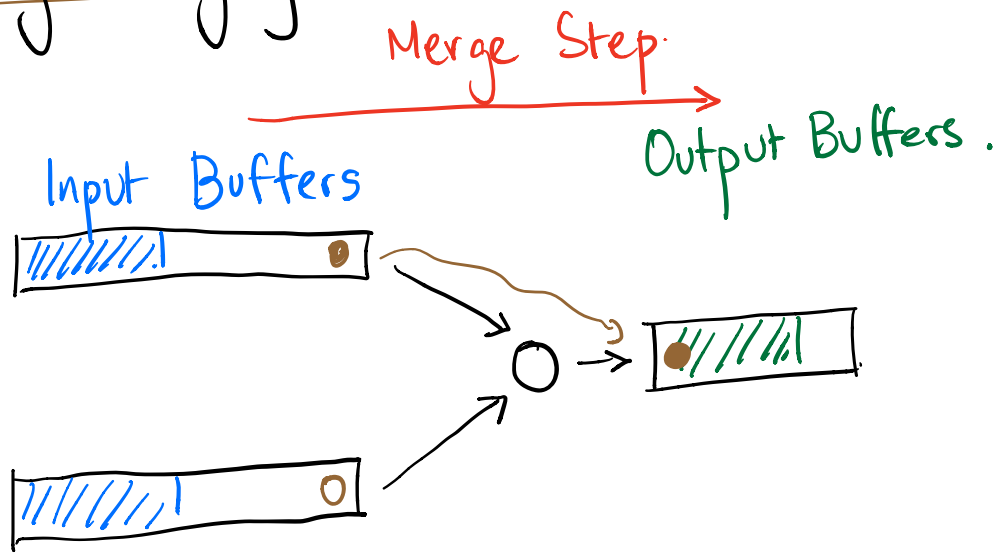
Funnel sorting time:

$$\Theta\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$$

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

FUNNEL - SORTING : A RECAP

Binary Merging



Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory, B blocks.

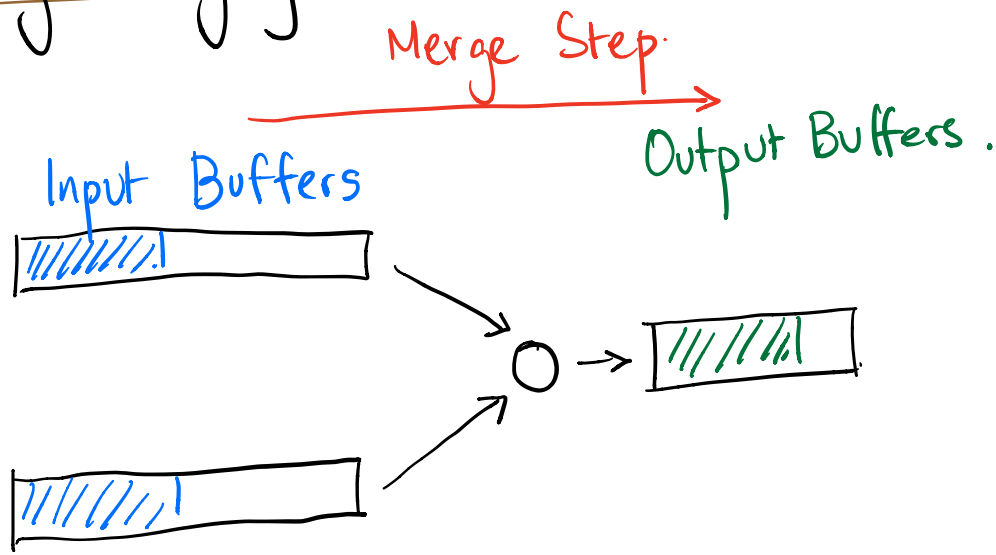
Funnel sorting time:

$$\Theta\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$$

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

FUNNEL - SORTING : A RECAP

Binary Merging



Idea: Chain together to a tree

Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory
B blocks.

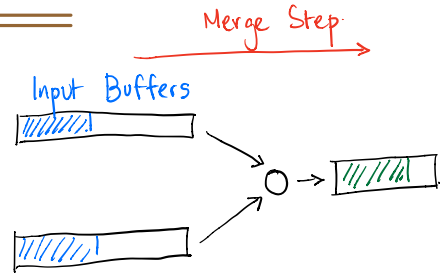
Funnel sorting time:

$$\Theta\left(\frac{N}{B} \log_{M/B} N/B\right)$$

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

FUNNEL - SORTING : A RECAP

Binary Merging



Thm 1 (Cache-Oblivious Sorting)

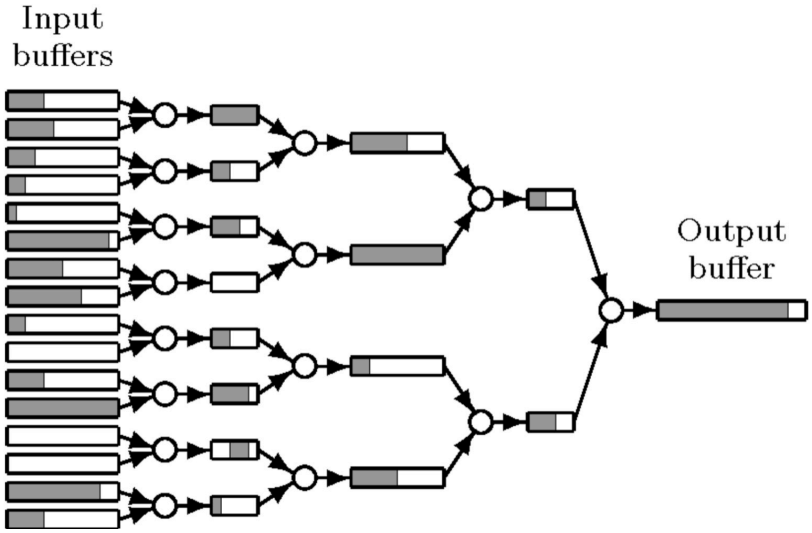
N items, M (small) memory

B blocks.

Funnel sorting time:

$$\Theta\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$$

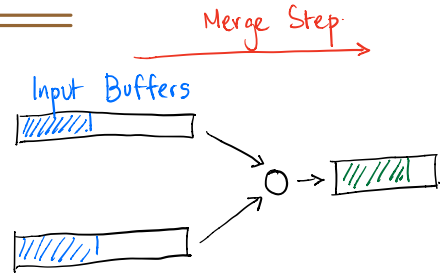
* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

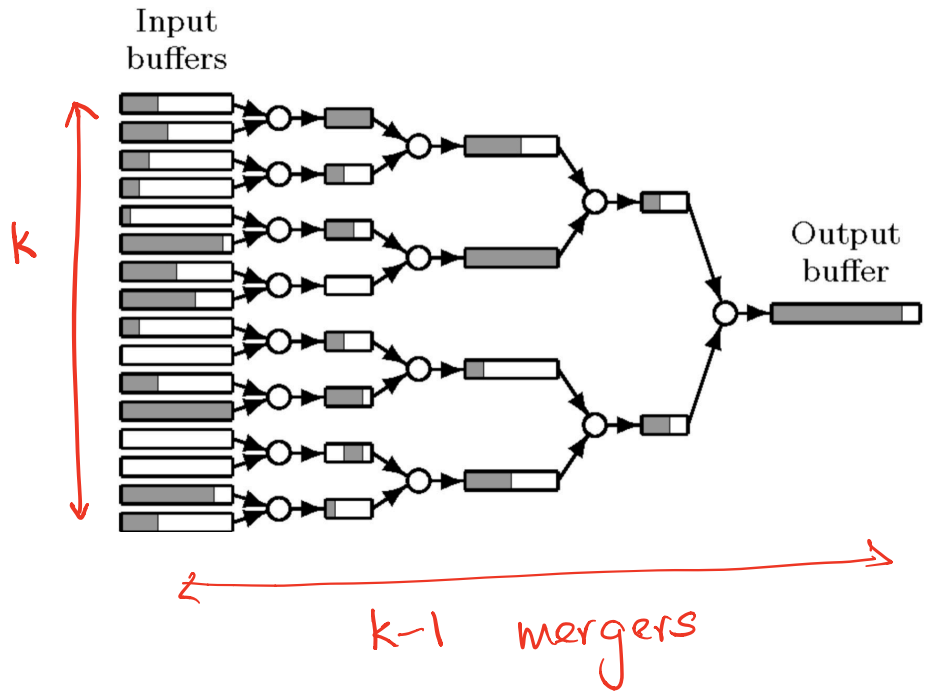


Tree

FUNNEL - SORTING : A RECAP

Binary Merging





Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory, B blocks.

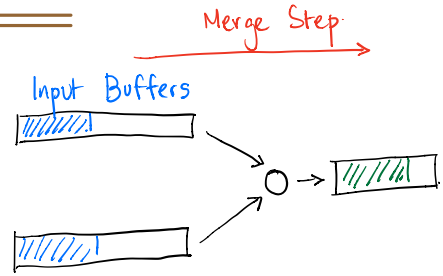
Funnel sorting time:

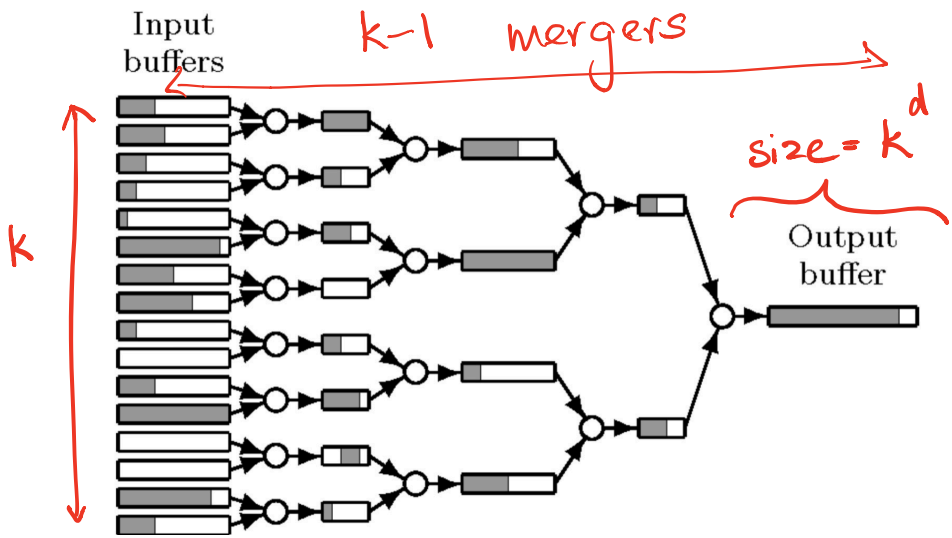
$$\Theta\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$$

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

FUNNEL - SORTING : A RECAP

Binary Merging





Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory, B blocks.

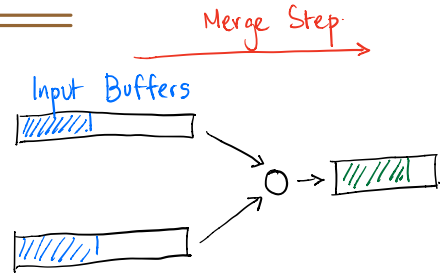
Funnel sorting time:

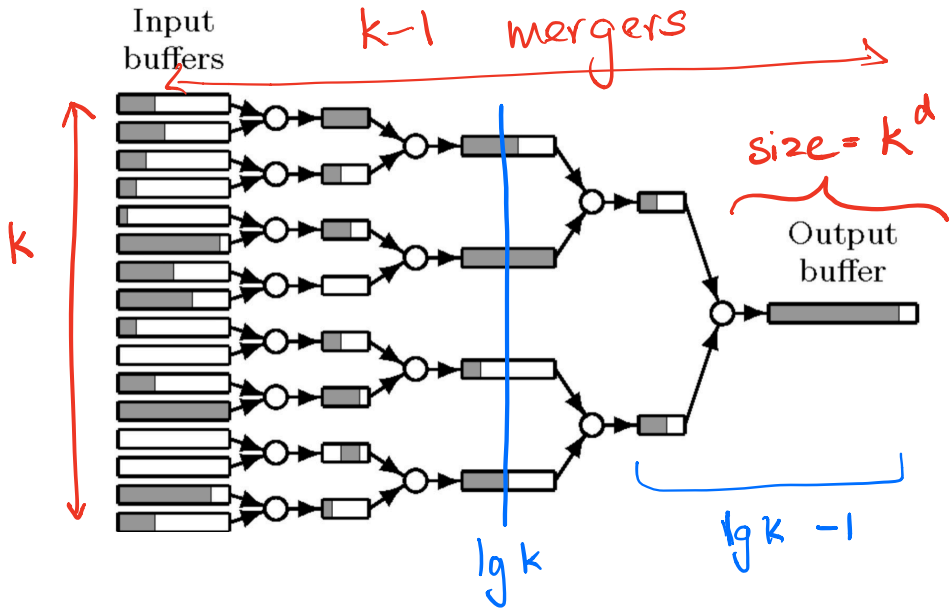
$$\Theta\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$$

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

FUNNEL - SORTING : A RECAP

Binary Merging





Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory, B blocks.

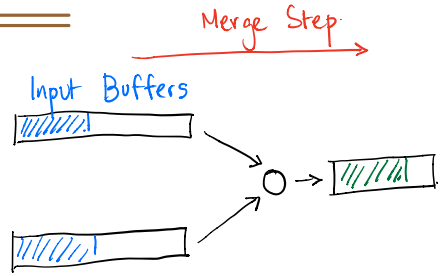
Funnel sorting time:

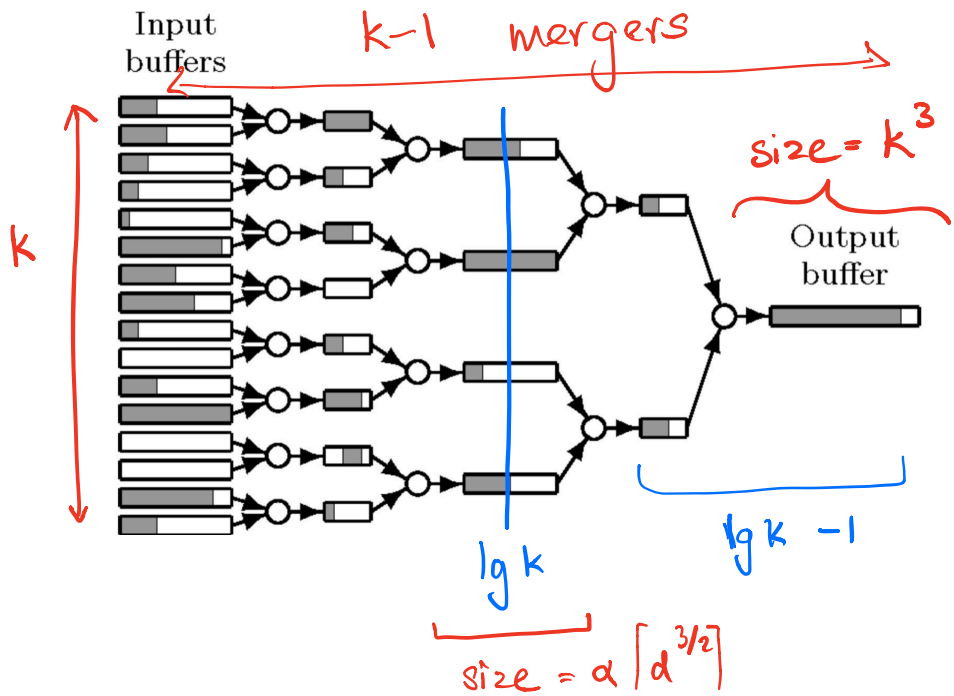
$$\Theta\left(\frac{N}{B} \log_{M/B} N/B\right)$$

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

FUNNEL - SORTING : A RECAP

Binary Merging





Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory, B blocks.

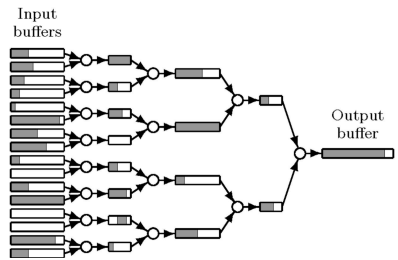
Funnel sorting time:

$$\Theta\left(\frac{N}{B} \log_{M/B} N/B\right)$$

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

FUNNEL-SORTING : A RECAP

Lazy Merging



Merges will be lazy:

Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory, B blocks.

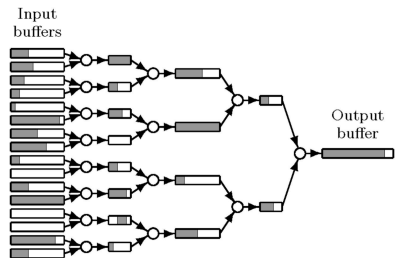
Funnel sorting time:

$$\Theta\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$$

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

FUNNEL-SORTING : A RECAP

Lazy Merging



Merges will be lazy:

Procedure FILL(v)
  while v's output buffer is not full
    if left input buffer empty
      FILL(left child of v)
    if right input buffer empty
      FILL(right child of v)
    perform one merge step

Fig. 2. The merging algorithm.

Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory, B blocks.

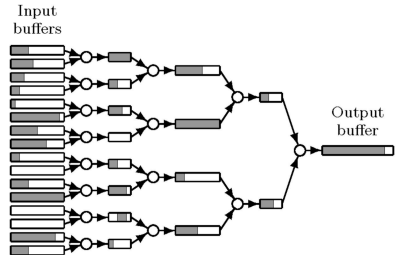
Funnel sorting time:

$$\Theta\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$$

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

FUNNEL-SORTING : A RECAP

Lazy Merging



Merges will be lazy:

lazy: fill up an buffer only if it runs empty. (avoid silly checks)

Procedure FILL(v)
  while v's output buffer is not full
    if left input buffer empty
      FILL(left child of v)
    if right input buffer empty
      FILL(right child of v)
  perform one merge step

Fig. 2. The merging algorithm.

Thm 1 (Cache-Oblivious Sorting)
N items, M (small) memory
B blocks.
Funnel sorting time:

$$\Theta\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$$

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

Funnel-Sorting : An engineer's view

Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory, B blocks.

Funnel sorting time:

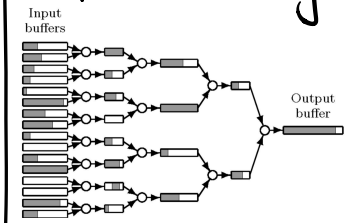
$$\Theta\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$$

Funnel Sorting

Primitives:

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

funnel sorting:



FUNNEL - SORTING : AN ENGINEER'S VIEW

Layout:
* DFS
* BFS
* vEB

Primitives:
* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

Funnel Sorting

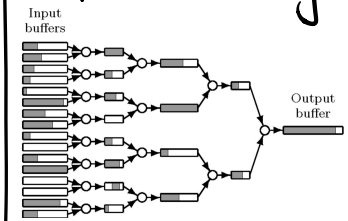
Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory, B blocks.

Funnel sorting time:

$$\Theta\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$$

funnel sorting:



Funnel-Sorting : An engineer's view

Layout:
* DFS
* BFS
* vEB

Merging:
* naive
* Hwang, Lin, Knuth
* Hybrid

Funnel Sorting

Primitives:
* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

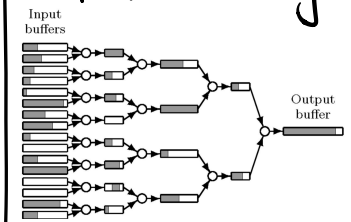
Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory, B blocks.

Funnel sorting time:

$$\Theta\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$$

funnel sorting:



FUNNEL - SORTING : AN ENGINEER'S VIEW

Layout:
* DFS
* BFS
* vEB

Merging:
* naive
* Hwang, Lin, Knuth
* Hybrid

Primitives:
* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

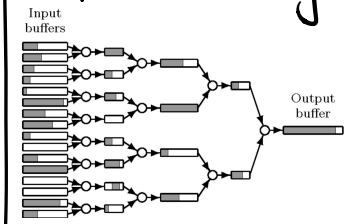
Funnel Sorting

B-way merges
* value of B

Thm 1 (Cache-Oblivious Sorting)
N items, M (small) memory
B blocks.
Funnel sorting time:

$$\Theta\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$$

funnel sorting:



FUNNEL-SORTING : AN ENGINEER'S VIEW

Thm 1 (Cache-Oblivious Sorting)

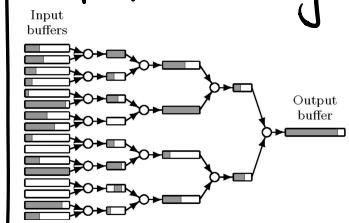
N items, M (small) memory

B blocks.

Funnel sorting time:

$$\Theta\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$$

funnel sorting:



Layout:
* DFS
* BFS
* vEB

Merging:
* naive
* Hwang, Lin, Knuth
* Hybrid

Primitives:
* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

Funnel Sorting

B-way merges
* value of B

Base Sort
* Insertion
* Selection
* Heap Sort
* Shell sort
* Quick sort
* std::sort

Hyper parameters
* α, d in buffer sz. αk^d

FUNNEL-SORTING : AN ENGINEER'S VIEW

Idea: Try Everything !!

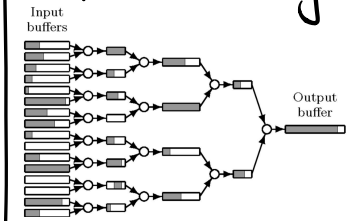
Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory, B blocks.

Funnel sorting time:

$$\Theta\left(\frac{N}{B} \log_{M/B} N/B\right)$$

funnel sorting:



FUNNEL - SORTING : AN ENGINEER'S VIEW

Method:

* Try different processors: { Pentium 3+4, MIPS, AMD, Itanium }
* data distributions
* data types:
  - ⇒ ints
  - ⇒ ints + ptrs
  - ⇒ records (100 bytes)
* randomness:
  - ⇒ drand48 → C lib

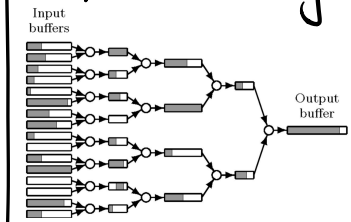
Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory, B blocks.

Funnel sorting time:

$$\Theta\left(\frac{N}{B} \log_{M/B} N/B\right)$$

funnel sorting:



Funnel-Sorting : An engineer's view

Thm 1 (Cache-Oblivious Sorting)

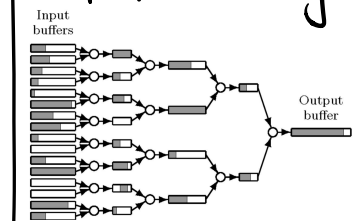
N items, M (small) memory

B blocks.

Funnel sorting time:

$$\Theta\left(\frac{N}{B}\log_{M/B} N/B\right)$$

funnel sorting:



Funnel Sorting

Layout:
* DFS
* BFS
* vEB

Merging:
* naive
* Hwang, Lin, Knuth
* Hybrid

Primitives:
* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

B-way merges
* value of B

Base Sort
* Insertion
* Selection
* Heap Sort
* Shell sort
* Quick Sort
* std::sort

Hyper parameters
* α, d in buffer sz. αk^d

Funnel-Sorting : An engineer's view

Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory, B blocks.

Funnel sorting time:

$$\Theta\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$$

Layout:

* DFS
* BFS
* vEB

winner: ~65%
vEB + std.allocator + nodes and buffers separate

Merging:

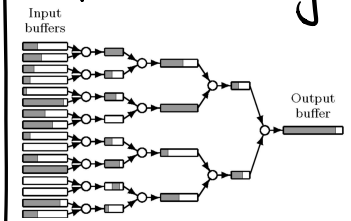
* naive
* Hwang, Lin, Knuth
* Hybrid

Primitives:

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

Funnel Sorting

funnel sorting:



B-way merges

* value of B

Base Sort

* Insertion
* Selection
* HeapSort
* Shell sort
* Quick sort
* std::sort

Hyper parameters

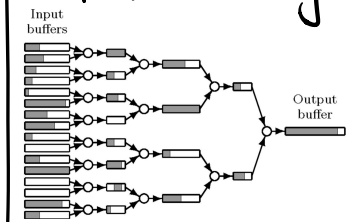
* α, d in buffer sz. αk^d

Funnel-Sorting : An engineer's view

Thm 1 (Cache-Oblivious Sorting)
N items, M (small) memory
B blocks.
Funnel sorting time:

$$\Theta\left(\frac{N}{B}\log_{M/B} N/B\right)$$

funnel sorting:



Layout:
* DFS
* BFS
* vEB

Merging: ~15-45%.
* naive
* Hwang, Lin, Knuth
* Hybrid

Winner:
naive
(bc branch) pred.

Primitives:
* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

Funnel Sorting

B-way merges
* value of B

Base Sort
* Insertion
* Selection
* HeapSort
* Shell sort
* Quick sort
* std::sort

Hyper parameters
* α, d
in buffer sz. αk^d

Funnel-Sorting : An engineer's view

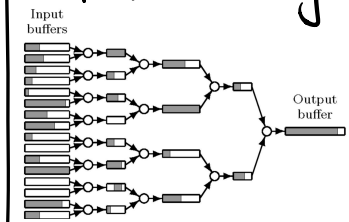
Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory, B blocks.

Funnel sorting time:

$$\Theta\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$$

funnel sorting:



Layout:
* DFS
* BFS
* vEB

Merging:
* naive
* Hwang, Lin, Knuth
* Hybrid

Primitives:
* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

Funnel Sorting

B-way merges (~40-65%)
* value of B
min: 4-5

Base Sort
* Insertion
* Selection
* Heap Sort
* Shell sort
* Quick Sort
* std::sort

Hyper parameters
* α, d
in buffer sz. αk^d

FUNNEL - SORTING : AN ENGINEER'S VIEW

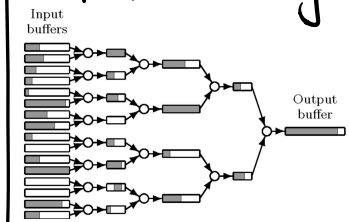
Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory, B blocks.

Funnel sorting time:

$$\Theta\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$$

funnel sorting:



Funnel Sorting

Layout:
* DFS
* BFS
* vEB

Merging:
* naive
* Hwang, Lin, Knuth
* Hybrid

Primitives:
* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

B-way merges
* value of B

Base Sort
* Insertion
* Selection
* Heap Sort

Winner: std::sort (before 400)
* Shell sort
* Quick sort
* std::sort

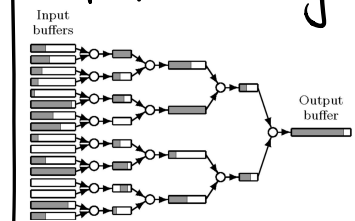
Hyper parameters
* α, d in buffer sz. αk^d

Funnel-Sorting : An engineer's view

Thm 1 (Cache-Oblivious Sorting)
N items, M (small) memory
B blocks.
Funnel sorting time:

$$\Theta\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$$

funnel sorting:



Layout:
* DFS
* BFS
* vEB

Merging:
* naive
* Hwang, Lin, Knuth
* Hybrid

Primitives:
* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

Funnel Sorting

B-way merges
* value of B

Base Sort
* Insertion
* Selection
* Heap Sort
* Shell sort
* Quick Sort
* std::sort

Hyper parameters
* α, d in buffer sz. αk^d

Winner: (~4x)
$\alpha = 16$
$k = 2$

Funnel-Sorting : An engineer's view

Layout:
* ~~DFS~~
* ~~DFS~~
* vEB

Merging:
* naive
* ~~Hwang, Lin, Knuth~~
* ~~Hybrid~~

Primitives:
* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

Funnel Sorting

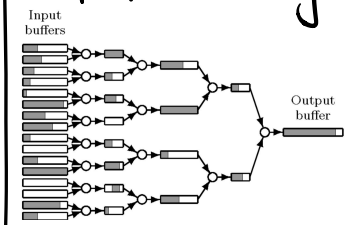
B-way merges
* value of B
4

Base Sort
* ~~Insertion~~
* ~~Selection~~
* ~~Heap Sort~~
* ~~Shell sort~~
* ~~Quick Sort~~
* std::sort

Hyper parameters
* α, d
$(16, 2)$ in buffer sz. αk^d

Thm 1 (Cache-Oblivious Sorting)
N items, M (small) memory
B blocks.
Funnel sorting time:
$$\Theta\left(\frac{N}{B} \log_{M/B} N/B\right)$$

funnel sorting:



FUNNEL-SORTING : AN ENGINEER'S VIEW

Thm 1 (Cache-Oblivious Sorting)

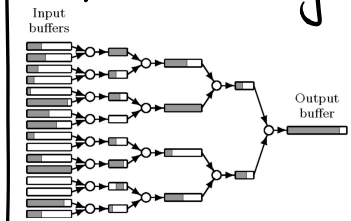
N items, M (small) memory

B blocks.

Funnel sorting time:

$$\Theta\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$$

funnel sorting:



Vs Competitors:

* GCC std::sort
* Intel Dinkumware std::sort
* Bentley's sort
* (Cache-aware) TPIE sort
* (Cache-aware) Xiao et.al sort
* (Cache-aware) Arge et.al sort

FUNNEL - SORTING : AN ENGINEER'S VIEW

Thm 1 (Cache-Oblivious Sorting)

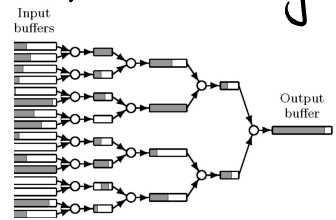
N items, M (small) memory

B blocks.

Funnel sorting time:

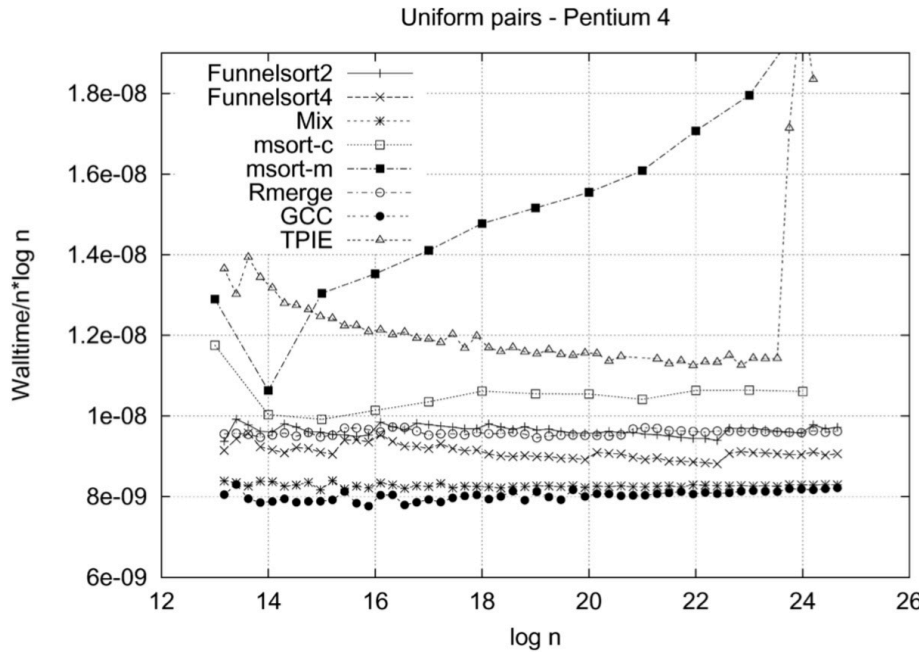
$$\Theta\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$$

funnel sorting:



Vs Competitors:

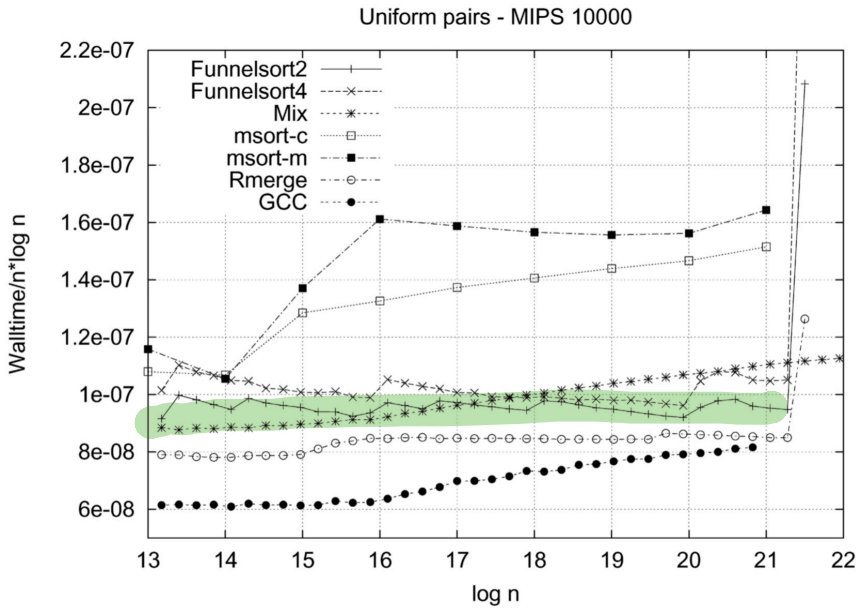
RAM:



FUNNEL - SORTING : AN ENGINEER'S VIEW

Vs Competitors:

RAM:



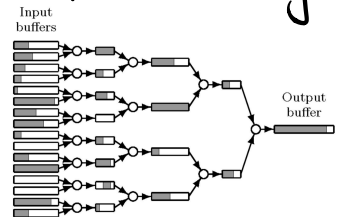
Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory, B blocks.

Funnel sorting time:

$$\Theta\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$$

funnel sorting:



FUNNEL - SORTING : AN ENGINEER'S VIEW

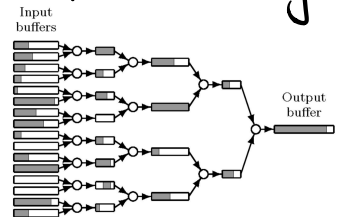
Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory, B blocks.

Funnel sorting time:

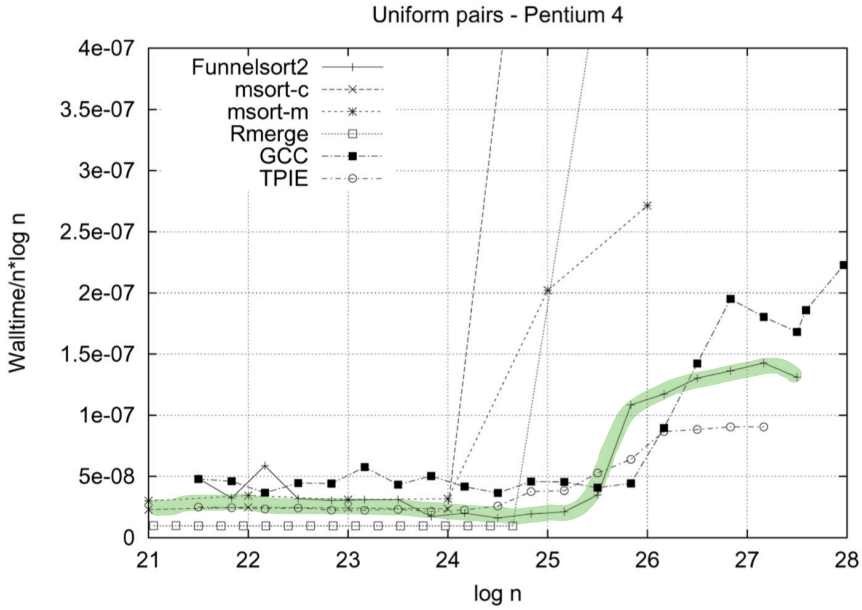
$$\Theta\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$$

funnel sorting:



Vs Competitors:

Disk:



FUNNEL - SORTING : AN ENGINEER'S VIEW

Thm 1 (Cache-Oblivious Sorting)

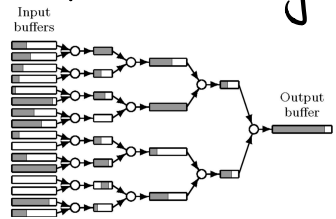
N items, M (small) memory

B blocks.

Funnel sorting time:

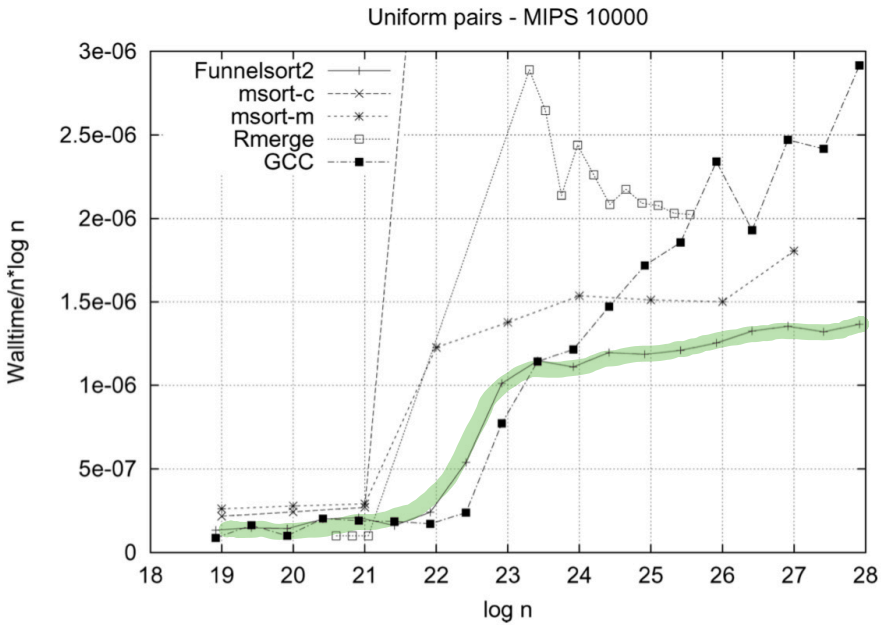
$$\Theta\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$$

funnel sorting:



Vs Competitors:

Disk:



Cache-Oblivious Sorting: Discussion Questions.

Discussion: (Feel free to also propose)

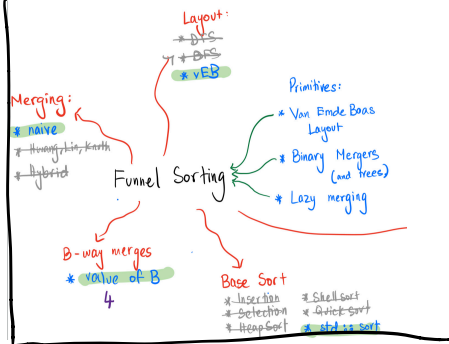
* Should we do (hard-core ablation-style) experiments for every algorithm?
* Does ablation style even work?
* Can these funneling ideas generalize to other interesting problems? cache-oblivious problems?
* Other architectures than a CPU

Thm 1 (Cache-Oblivious Sorting)

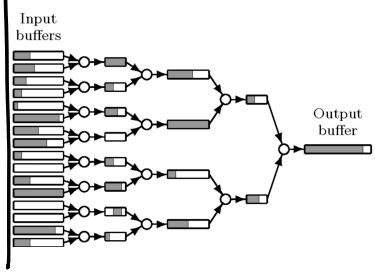
N items, M (small) memory, B blocks.

Funnel sorting time:

$$\Theta\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$$



Funnel sorting:



6.886 talk: Engineering a cache-oblivious sorting algorithm

Brodal, Fagerberg, Vinther (2008)

Sualeh Asif

MIT Computer Science Artificial Intelligence Laboratory
Massachusetts Institute of Technology

